UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/518,974	12/21/2004	Richard Michael Taylor	5035-202US//P29,653	2918

20802 7590 02/28/2008

SYNNESTVEDT LECHNER & WOODBRIDGE LLP
P O BOX 592
112 NASSAU STREET
PRINCETON, NJ 08542-0592

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
02/28/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/518,974	TAYLOR, RICHARD MICHAEL	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 November 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-34* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7 and 9-34* is/are rejected.

7)☒ Claim(s) *8* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 December 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-34 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: RCE and Amendment as received on 11/19/2007.

Maintained Rejections

3. Applicant has failed to overcome the prior art rejections set forth in the previous Office Action. Consequently, these rejections are respectfully maintained by the examiner and are copied below for applicant's convenience.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-5, 9-11, 23, 25-27, 30-32, and 34 are rejected under 35 U.S.C. 102(b) as being anticipated by Morrison et al., U.S. Patent No. 4,847,755 (herein referred to as Morrison).

6. Referring to claim 1, Morrison has taught a method of instruction execution within a microprocessor whereby:

(a) a sequence of operations from a single execution thread is divided into individual strands. See Fig.5, and note that a strand may be considered an individual basic block (BBn) or a group of basic blocks (ESn). The strands make up a single thread as Morrison is not a multithreaded processor.

(b) the strands are numbered at compile time to provide an implicit logical time ordering. See Fig.5 and column 4, lines 42-53, and note the numbering of the strands, which is done prior to run-time (at compile-time, where compile-time is interpreted as the time before run-time).

(c) the operations within each individual strand are explicitly labeled with strand numbering and are executed sequentially. See Table 5 in column 12. Note that each instruction is explicitly labeled with an instruction firing time (IFT), which is the time at which the associated instruction is selected for execution. The instructions are also executed sequentially, i.e., one after another.

(d) certain operations from different strands may be executed out-of-order with respect to their original sequential order. Again, see Table 5 in column 12 as well as Table 1 in column 8. Table 1 shows six instructions of a strand in original order. However, Table 5 shows that these same instructions are able to execute out of order.

(e) each strand has a predication state that determines whether certain operations from the strand should be completed. See column 8, Table 1, and column 37, line 62, to column 38, line 12. Note that all strands may include a conditional branch at the end (as shown in Table 1). This conditional branch specifies a condition. If the condition is met, then a first path of instructions will be executed and a second path will not be executed. For instance, Table 1 shows a loop, and when the condition is met in the branch, the system will branch back to I0 and repeat the instructions. Consequently, it can be seen that the (N+1)th execution of I0-I4 is predicated on

the Nth execution of I5. That is, if the condition of I5 in the Nth iteration is met, then it is determined that I0-I4 should be completed in the (N+1)th iteration. However, if the condition of I5 in the Nth iteration is not met, then it is determined that I0-I4 should not be completed in the (N+1)th iteration.

7. Referring to claim 2, Morrison has taught a method as described in claim 1. Morrison has further taught that operations in one operation strand are able to modify the predication status of another strand. See column 8, Table 1, and note that if the condition set forth by the branch instruction (I5) is met, then a subsequent strand will not be completed (since the system will loop back to the beginning of the current strand). However, if the condition is not met, then a subsequent strand would be entered and completed. Consequently, it can be seen that completion of operations in a second strand is predicated on operations in a first strand.

8. Referring to claim 3, Morrison has taught a method as described in claim 1. Morrison has further taught that a plurality of strands are further composed into an executable code block. Strands inherently include instructions and groups of instructions form an executable code block. Also, with the interpretation that basic blocks are strands, as shown in Fig.5, basic blocks are grouped to form execution sets (ESn), i.e., executable code blocks.

9. Referring to claim 4, Morrison has taught a method as described in claim 3. Morrison has further taught that means are provided to reset the predication status of each individual strand at the start of the execution of the code block. The examiner deems this to be inherent because when the system first begins execution of each strand, the instructions of that strand will complete. Only when the last instruction of the strand (Table 1) is encountered will the

predication status possibly be modified (because the last instruction controls whether the strand continues or not).

10. Referring to claim 5, Morrison has taught a method as described in claim 4. Morrison has further taught that certain operation strands can be given an abort status indicating that certain operations in the block could not be completed. Again, see Table 1 in column 8. If the branch condition is not met, then the strand is "given abort status" by indicating that the operations of that strand should not be completed in the next iteration.

11. Referring to claim 9, Morrison has taught a method as described in claim 3. Morrison has further taught that the subdivision of code into strands is performed from an original sequential stream of instructions. See Fig.5. Since strands comprise sequential instructions from a single thread, strands are formed from an original stream of instructions.

12. Referring to claim 10, Morrison has taught a method as described in claim 3. Morrison has further taught that the subdivision of code into executable blocks is performed from an original sequential stream of instructions. See Fig.5. Since blocks comprise sequential instructions from a single thread, blocks are formed from an original stream of instructions.

13. Referring to claim 11, Morrison has taught a method as described in claim 3. Morrison has further taught that each operation strand is subdivided into a number of phases according to the type of operations that may be issued and operations that modify processor state that is visible outside of the executable block may only be executed in the final phase of each strand. Each strand can be seen as being divided into a number of phases. Since all strands end in a branch instruction (column 7, lines 18-36), then each strand has a pre-branch phase and a branch-

phase. A branch instruction modifies processor state such that it controls which path the processor will execute, and it executes in a final phase of the strand (the branch phase).

14. Referring to claim 23, Morrison has taught a method as described in claim 3. Morrison has further taught that each strand has a committed phase and entry to the committed phase of each strand is performed in the implicit logical time ordering of the strands. The examiner deems this inherent as, in general, strands executed after previous strands will be committed after the previous strands are committed. That is, the very first strand is not executed and then held off from being committed until after every other strand is committed. This ensures proper program execution.

15. Referring to claim 25, Morrison has taught a method as described in claim 23. Morrison has further taught that a branch executed from a particular operation strand causes all strands which are logically later to be disabled. See Table 1 in column 8. If the branch (I5) sets forth a condition which is met, then execution will continue at I0 (loop), and consequently, all later strands will not execute (be disabled).

16. Referring to claim 26, Morrison has taught a method as described in claim 25. Morrison has further taught that branches may be issued out of their original sequential order but are suitably resolved and no actual branch is performed until the end of the executable block is reached. This is the idea behind out-of-order processing. A branch may be the 100^{th} instruction in the program but not be the 100^{th} instruction executed. Furthermore, a branch always ends an executable block as it branches to a new block, so a branch is not performed until the end of the block (see column 8, Table 1).

17. Referring to claim 27, Morrison has taught a method as described in claim 1. Morrison has further taught that operations from different strands may be interspersed in the execution word for the purposes of improving code density. See Fig.5 and note that in a single execution word (ESn), multiple strands exist, and therefore, operations from different strands are mixed.

18. Referring to claim 30, Morrison has taught a method as described in claim 3. Morrison has further taught that the execution status of strands upon entry to the executable block may set from a parameter provided by a preceding branch operation. A conditional branch contains a parameter (condition field) which, depending on the value, cause a target strand of instructions to be skipped or executed, and a fall-through strand to be skipped or executed. That is, if the branch is to be taken, depending on its condition code, then the execution status of the target strand is "execute" while the execution status of the fall-through strand is "don't execute".

19. Referring to claim 31, Morrison has taught a method as described in claim 30. Morrison has further taught that an entry mechanism may be used to reposition a branch in a logically later strand in the block. See column 38, lines 56-57, and note the use of delayed branching, which essentially means that branches are repositioned such that delay slot instructions are executed after the branch. For instance, assume a branch exists at instruction 10 in a logically later strand. If there are two delay slots and two instructions are identified as delay slot instructions before the branch, then the branch will effectively be repositioned to instruction 8 while the two delay slot instructions become instructions 9 and 10.

20. Referring to claim 32, Morrison has taught a method as described in claim 1. Morrison has further taught that the scheduling and construction of strands is influenced by profiling of the

code. See the abstract. Essentially, the strands are scheduled and built based on at least dependencies.

21. Referring to claim 34, Kadambi has taught a microprocessor adapted to execute instructions using the method of claim 1. See the claim 1 rejection.

Claim Rejections - 35 USC § 103

22. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

23. Claims 6-7, 24, 28-29, and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morrison.

24. Referring to claim 6, Morrison has taught a method as described in claim 5. Morrison has not taught that the abort status mechanism may be used to support the recovery from data speculative operations between strands. Specifically, Morrison has not taught speculation. However, Official Notice is taken that speculation (branch prediction) is well known and accepted in the art. Speculation allows the system to speculate the outcome of a branch before it is known so that it can begin execution of subsequent instructions sooner. If the speculation turns out to be correct, then performance has been improved. If, however, speculation turns out to be incorrect, then the already begun subsequent instructions must be aborted/flushed so that the correct instructions are able to execute instead. Speculation is a known technique for improving performance in a system with conditional branches, which Morrison includes.

Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison to include speculation between strands and recovery from incorrect speculation to ensure correct execution.

25. Referring to claim 7, Morrison has taught a method as described in claim 6. Morrison, as modified, has further taught that execution of the code block should be repeated when an abort occurs. Taking Table 1 in column 8 as an example, when the branch I5 is encountered, if the system predicts that the branch will be not taken, then instructions I0-I4 will no longer execute, and execution will proceed to the next strand. However, if this prediction turns out to be incorrect, the next strand's operations will be aborted and execution of I0-I4 will be repeated.

26. Referring to claim 24, Morrison has taught a method as described in claim 23. Morrison has not taught that an abort of a certain operation strand also causes an abort of all strands which are logically later. However, Official Notice is taken that speculation (branch prediction) is well known and accepted in the art. Speculation allows the system to speculate the outcome of a branch before it is known so that it can begin execution of subsequent instructions sooner. If the speculation turns out to be correct, then performance has been improved. If, however, speculation turns out to be incorrect, then the already begun subsequent instructions must be aborted/flushed so that the correct instructions are able to execute instead. Speculation is a known technique for improving performance in a system with conditional branches, which Morrison includes. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison to include speculation between strands and recovery from incorrect speculation to ensure correct execution. It follows that if a branch is

mispredicted, then all instructions along the wrong path, whether in the same strand or logically later strand, will have to be aborted.

27. Referring to claim 28, Morrison has taught a method as described in claim 1. Morrison has not taught that an explicit operation may be issued that disables a set of strands depending on whether they are logically the first being executed. However, Official Notice is taken that speculation (branch prediction) is well known and accepted in the art. Speculation allows the system to speculate the outcome of a branch before it is known so that it can begin execution of subsequent instructions sooner. If the speculation turns out to be correct, then performance has been improved. If, however, speculation turns out to be incorrect, then the already begun subsequent instructions must be aborted/flushed so that the correct instructions are able to execute instead. Speculation is a known technique for improving performance in a system with conditional branches, which Morrison includes. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison to include speculation between strands and recovery from incorrect speculation to ensure correct execution. It follows that, in the case of a mispredicted branch or an overeager load, for instance, the strands that are first executed following those instructions will be canceled (flushed) or replayed. Therefore, a flush operation or replay operation would be issued.

28. Referring to claim 29, Morrison has taught a method as described in claim 10. Morrison has not taught that an operation strand mechanism is used to convert conditional blocks of code constructed using branches into separate operations strands that do not require a branch. However, Official Notice is taken that the if-conversion and its benefits are well known and accepted in the art. That is, if-conversion is a process in which all branches are replaced with

predication. Or, control dependence is changed into data dependence, which means that the code will no longer need to be executed using branch prediction which is time costly if there is ever a misprediction. Instead, since operation execution merely depends on its predicate and its predicate is set by some preceding instruction, the instructions can simply be scheduled like every other out of order operation. There is no misprediction penalty to worry about, and furthermore, there would be no need for branch prediction hardware. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison such that branches could be replaced with predicated code.

29. Referring to claim 33, Morrison has taught a method as described in claim 32. Morrison has not taught that a strand ordering is used to implement static speculations that provides performance benefit whilst seeking to minimize the chances of an incorrect speculation that requires recovery. However, Official Notice is taken that speculation (branch prediction) is well known and accepted in the art. Speculation allows the system to speculate the outcome of a branch before it is known so that it can begin execution of subsequent instructions sooner. If the speculation turns out to be correct, then performance has been improved. If, however, speculation turns out to be incorrect, then the already begun subsequent instructions must be aborted/flushed so that the correct instructions are able to execute instead. Speculation is a known technique for improving performance in a system with conditional branches, which Morrison includes. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison to include speculation between strands and recovery from incorrect speculation to ensure correct execution. And, it follows that reducing

incorrect speculations it is inherent in any speculative, out of order machine as clearly the goal is to increase performance yet minimize costly incorrect speculation recovery.

30. Claims 12-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morrison in view of Asato, U.S. Patent No. 6,289,442 (as applied in the previous Office Action).

31. Referring to claim 12, Morrison has taught a method as described in claim 3. Morrison has not taught that operations from the strand may be tagged within their execution word format to indicate the strand to which they belong. However, Asato has taught such a concept. See Fig.6B. Note that operations are tagged to indicate which strand they belong to. This way, depending on a conditional branch outcome, the tag can be broadcasted to ensure abortion of the incorrect strand. Therefore, in order to assist in invalidating instructions that are incorrectly executing, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison to include execution word format tags for operations.

32. Referring to claim 13, Morrison in view of Asato has taught a method as described in claim 12. Asato has further taught that a tagged strand number is utilized in the control logic of the functional unit to affect the execution of the operation. See the abstract and Fig.7. Essentially, the tag of the correct path is broadcasted and compared with the tags of the two paths being executed. The strand having the tag that does not match the broadcasted tag is invalidated.

33. Referring to claim 14, Morrison in view of Asato has taught a method as described in claim 13. Asato has further taught that execution from a disabled strand substantially disables the operation or prevents writeback of results that could affect processor state. See the abstract and Fig.7.

34. Referring to claim 15, Morrison in view of Asato has taught a method as described in claim 14. Asato has further taught that the tagging of operations may be selective and need only necessarily apply to operations that affect processor state that is visible outside of the executable block. See Fig.6B. The tagging is selective, as the tag values can be assigned according to a user-specified algorithm. In addition, since all instruction affect a processor state, all operations are tagged.

35. Referring to claim 16, Morrison in view of Asato has taught a method as described in claim 14. Morrison in view of Asato has further taught that the execution state of each operation strand is distributed to certain functional units by a global bus structure. As previously discussed, when misspeculation occurs, a flush/invalidate signal is sent to functional units. This signal represents the execution state of the strands in that it indicates that the units are in an incorrect execution state.

36. Referring to claim 17, Morrison in view of Asato has taught a method as described in claim 16. Morrison in view of Asato has further taught that the strand execution state is calculated and maintained in a strand control unit. This is deemed inherent as some subsystem must exist which will calculate when the execution is correct and when it is incorrect, and based on the state, the appropriate signals will be sent.

37. Referring to claim 18, Morrison in view of Asato has taught a method as described in claim 17. Morrison in view of Asato has further taught that the strand control unit receives requests to modify strand status from one or more functional units. The strand control unit can only indicate that execution needs to be repeated or flushed in response to a signal that misspeculation has occurred. Consequently, the strand control unit, after receiving a signal that

misspeculation has occurred (which inherently comes from some functional unit(s), will modify the strand status such that it must be repeated or flushed.

38. Claims 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morrison in view of Kadambi, U.S. Patent No. 7,055,021 (as applied in the previous Office Action).

39. Referring to claim 19, Morrison has taught a method as described in claim 7. Morrison has not taught that an abort mechanism is utilized to provide a load speculation mechanism allowing memory loads to be executed earlier than a logically preceding store operation that may access the same address. However, Kadambi has taught such a concept. See column 21, line 62, to column 22, line 4. Load boosting is a well-known concept which increase performance as it is out-of-order execution of memory operations. Out-of-order execution, which is known, has clear advantages. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison to include load speculation as taught by Kadambi.

40. Referring to claim 20, Morrison in view of Kadambi has taught a method as described in claim 19. Kadambi has further taught that the load speculation mechanism provides recovery from incorrect speculations by repeat execution of the executable block without the requirement for special compensation code. See column 21, line 56, to column 22, line 4. Incorrect speculations result in recovery (repeating execution). There is no mention of special compensation code to perform this. The instructions that need to be replayed are simply replayed.

41. Referring to claim 21, Morrison in view of Kadambi has taught a method as described in claim 19. Kadambi has further taught that detection of incorrect load speculations is performed

by an explicit functional unit that is used to compare the addresses being used by the load and store operations. See column 21, line 64, to column 22, line 4. If overeager loads need to be replayed, and overeager loads have the same address as an older store, then it is inherent that some comparison of addresses is performed by a functional unit to determine that the addresses are the same.

42. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Morrison in view of Kadambi and further in view of Hastings, U.S. Patent No. 5,193,180 (as applied in the previous Office Action).

43. Referring to claim 22, Morrison in view of Kadambi has taught a method as described in claim 21. Morrison in view of Kadambi has not taught that address checks are inserted into the code strands as a result of insertion of such operations within a graph representation of the strand built at code generation time. However, Hastings has taught such a concept. See column 3, lines 29-55, and note that prior to memory operations in strands, address check instructions are inserted which allow the system to detect array boundary violations and similar data errors. As a result, in order to ensure that memory access violations are avoided in Kadambi, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morrison in view of Kadambi to include the insertion of address checks in the code, as taught by Hastings.

Allowable Subject Matter

44. Claim 8 is objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Response to Arguments

45. Applicant's arguments filed on November 19, 2007, have been fully considered but they are not persuasive.

46. Applicant argues the novelty/rejection of claim 1 on page 8 of the remarks, in substance that:

> "The Examiner's reliance on Morrison to reject the claims is based on the incorrect analogizing of the claimed "strand numbering" to the "instruction numbering" of Morrison. For instance, the examiner draws an equivalence between the claimed strand numbering and the instruction numbering of Table 5 in column 12 of Morrison. However, they are not the same; Morrison's numbering is of individual instructions within each basic block whereas the strand number is a number of multiple instructions across basic blocks. Thus, with the present claimed invention multiple instructions (typically within the same basic block) can be given the same strand number, whereas in Morrison's approach each individual instruction is given a different number. This is a key difference as the claimed strand approach provides a mechanism to label multiple instructions with a common conditionality or predicate status. This is neither taught nor suggested by Morrison."

47. These arguments are not found persuasive for the following reasons:

a) The examiner asserts that applicant has not defined "strand numbering" in the claims. Consequently, the examiner is to give the broadest reasonable interpretation of "strand numbering". The numbering shown in Table 5 of Morrison certainly qualifies as strand numbering because the numbering is applied to instruction in strands.

48. Applicant argues the novelty/rejection of claim 1 on page 9 of the remarks, in substance that:

> "The Examiner also states that operations in Morrison may be performed out of order with respect to their numbering. This is indeed the case, but in Morrison this corresponds to out of order instruction execution within a basic block. In the present claimed invention (since strand numbers are typically associated with different basic blocks) this corresponds to out of order issue both within and across basic blocks. In particular, Morrison, column 20, lines 15-25, describes how instructions in different basic blocks are specifically labeled with disjoint numbers so that there is no mixing of instructions across the basic blocks. The strand numbering of the present claimed invention specifically, and advantageously, allows mixing of instructions across basic blocks, as it enables parallelism to be exploited over a wider program scope."

49. These arguments are not found persuasive for the following reasons:

a) The claimed out-of-order execution is not limited to instructions across basic blocks such that instruction mixing is achieved. The claim merely calls for certain operations from different strands being executed out of order. Hence, instructions from a first strand, such as those shown in Tables 1 and 5 of Morrison, may first be executed out of order, and then instructions from a second strand may be executed out of order. This type of execution reads on the current claim language.

50. Applicant argues the novelty/rejection of claim 1 on pages 9-10 of the remarks, in substance that:

> "The Examiner also states that instruction I5 from column 8, Table 1 is a branch that is used to determine the completion status for subsequent instructions. However this is not due to a predication state associated with the strand number (as claimed in the present invention) but because the subsequent instructions from the repeated basic block are not fetched if the branch is not taken. In other words, this approach of Morrison is a well known prior art control-based mechanism. The strand mechanism of the claimed invention is a data flow predication mechanism that associates a single predicate status with a previously labeled set of instructions sharing the same strand number. This is advantageous as it allows instructions from different strands (and thus basic blocks) to be mixed freely in the instruction schedule. For instance, if the loop of the Morrison example contained a conditionally executed instruction, then this would break the loop into multiple basic blocks. Instructions from the next basic block cannot be executed until the branch is resolved and all instructions from the previous basic block are

> completed (as discussed in the BEU delay mechanism of Column 13, lines 7 - 42). No parallelism exists between the basic blocks.
>
> In the approach of the present invention, by way of contrast, instructions with a strand number for the conditional basic block can be freely scheduled in time as long as the final predicate status for the strand is calculated prior to the execution of any instruction from the conditional strand that affects the final machine state. This capability greatly increases the scope for parallel instruction execution, and cannot be achieved by Morrison."

51. These arguments are not found persuasive for the following reasons:

a) Similar to previous arguments, applicant is arguing limitations which do not appear in the claims. Applicant's claim 1 merely calls for each strand having a predication state that determines whether operations from the strand should be completed. The branch instruction of Morrison, which is found at the end of every strand, makes such a determination for the reasons set forth in the rejection above. Applicant's claims are simply too broad.

Conclusion

52. All claims are drawn to the same invention claimed in the application prior to the entry of the submission under 37 CFR 1.114 and could have been finally rejected on the grounds and art of record in the next Office action if they had been entered in the application prior to entry under 37 CFR 1.114. Accordingly, **THIS ACTION IS MADE FINAL** even though it is a first action after the filing of a request for continued examination and the submission under 37 CFR 1.114. See MPEP § 706.07(b). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no, however, event will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/David J. Huisman/
Art Unit 2183
February 7, 2008

/Eddie P Chan/
Supervisory Patent Examiner, Art Unit 2183